Unsupervised Large Graph Embedding

Feiping Nie,[1] **Wei Zhu,**[1] **Xuelong Li**[2]
[1]School of Computer Science and Center for OPTical IMagery Analysis and Learning (OPTIMAL),
Northwestern Polytechnical University, Xi'an 710072, Shaanxi, P. R. China
[2]Center for OPTical IMagery Analysis and Learning (OPTIMAL),
Xi'an Institute of Optics and Precision Mechanics,
Chinese Academy of Sciences, Xi'an 710119, Shaanxi, P. R. China
{feipingnie@gmail.com, zwvews@gmail.com, xuelong_li@opt.ac.cn}

Abstract

There are many successful spectral based unsupervised dimensionality reduction methods, including Laplacian Eigenmap (LE), Locality Preserving Projection (LPP), Spectral Regression (SR), *etc*. LPP and SR are two different linear spectral based methods, however, we discover that LPP and SR are equivalent, if the symmetric similarity matrix is doubly stochastic, Positive Semi-Definite (PSD) and with rank p, where p is the reduced dimension. The discovery promotes us to seek low-rank and doubly stochastic similarity matrix, we then propose an unsupervised linear dimensionality reduction method, called Unsupervised Large Graph Embedding (ULGE). ULGE starts with similar idea as LPP, it adopts an efficient approach to construct similarity matrix and then performs spectral analysis efficiently, the computational complexity can reduce to $O(ndm)$, which is a significant improvement compared to conventional spectral based methods which need $O(n^2d)$ at least, where n, d and m are the number of samples, dimensions and anchors, respectively. Extensive experiments on several public available data sets demonstrate the efficiency and effectiveness of the proposed method.

Introduction

As one of the most efficient approach to deal with high-dimensional data, dimensionality reduction attracts many researchers' attentions, and lots of successful methods have been proposed. However, the excessive amounts of data bring lots of challenges, and make conventional methods inappropriate in real life application. The most popular spectral based unsupervised dimensionality reduction methods include Laplacian Eigenmap (LE) (Belkin and Niyogi 2001), Locality Preserving Projections (LPP) (He and Niyogi 2003), Spectral Regression (SR) (Cai, He, and Han 2007), etc. Spectral based methods always construct similarity matrix firstly, and then perform spectral analysis on the obtained matrix.

LPP and SR are two different linear methods, nevertheless, we discover that if the similarity matrix is symmetric, PSD, doubly stochastic and with rank p, LPP is equivalent to SR. As far as we know, this is the first research to discover the equivalence between these two popular methods, moreover, the discovery indicates that such specific similarity matrix has lots of advantages on both performance and efficiency, we thus prefer to construct such similarity matrix. However, conventional spectral methods always construct almost full-rank similarity matrix by k-nearest neighbor approach, which makes both graph construction as well as spectral analysis time consuming, and the time complexity is $O(n^2d)$ at least, it is surely unbearable for the data which contains hundreds of thousands samples.

To alleviate the problem, inspired by recent progresses on scalable semi-supervised learning (Liu, He, and Chang 2010), large scale spectral clustering (Cai and Chen 2015)(Li et al. 2015), and large scale spectral based dimensionality reduction (Cai 2015), we propose an efficient method, named Unsupervised Large Graph Embedding (ULGE). ULGE starts with similar idea as LPP, but benefits a lot from the the anchor-based graph strategy as well as the specific similarity matrix. The overall time complexity is $O(ndm)$, which has great advantage over conventional spectral based methods.

Three main contributions of this paper are listed as follows:

1. We discover that LPP is equivalent to SR under certain condition, which is of importance in guiding the efficient spectral based dimensionality reduction method design.
2. We propose a spectral based unsupervised linear dimensionality reduction method ULGE with a time complexity of $O(ndm)$. ULGE is efficient on both large graph construction and spectral analysis, which makes it suitable for large scale data sets.
3. Comprehensive experiments on several large scale data sets demonstrate the efficiency and effectiveness of the proposed method for dealing with large scale data.

We first introduce some notations that are used throughout the paper. For matrix $M \in \mathbb{R}^{r\times t}$, the (i,j)-th entry of M is denoted by m_{ij}, the transpose of the i-th row of M is denoted by $m_i \in \mathbb{R}^{t\times 1}$. Identity matrix is denoted by I. The trace of M is denoted by $Tr(M)$. The transpose of matrix M is denoted by M^T. The inverse of matrix M is denoted by M^{-1}. The F-norm of M is denoted by $\|M\|_F$. $\mathbf{1}$ is the column vector of all ones.

Background

In the past decades, many spectral based dimensionality reduction methods have been proposed. Most initial attempts are non-linear methods, e.g. LE (Belkin and Niyogi 2001),

LLE (T and K 2000), and ISOMAP (Tenenbaum 1997). Soon after, linear methods, e.g. LPP (He and Niyogi 2003) and SR (Cai, He, and Han 2007), have been proposed to deal with out-of-sample problem. There are kinds of linear dimensionality reduction methods, to sum these linear methods up , Yan *et al.* and Nie *et al.* propose two different linear dimensionality reduction frameworks, called graph embedding (Yan et al. 2007) and Flexible Manifold Embedding (FME) (Nie et al. 2010), respectively. Next, we briefly introduce some unsupervised dimensionality reduction methods which are most related to this paper.

Given a data matrix $X = [x_1, \ldots, x_n]^T \in \mathbb{R}^{n\times d}$, where $x_i \in \mathbb{R}^d$ denotes the i-th sample, let $A \in \mathbb{R}^{n\times n}$ be the similarity matrix constructed by k-nearest neighbor approach, and a_{ij} is the similarity between x_i and x_j. We now seek to find the low dimensional embedding, i.e. $Y \in \mathbb{R}^{n\times p}$, where p is the reduced dimension, LE performs dimensionality reduction via solving following problem (Belkin and Niyogi 2001):

$$\min_{Y} Tr((Y^T DY)^{-1} Y^T LY), \tag{1}$$

where $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix and the i-th entry is defined as $\sum_{j=1}^{n} a_{ij}$, $L = D - A$ denotes Laplacian matrix (Chung 1997). To keep things simple, Problem (1) is ratio trace representation (Wang et al. 2007). Problem (1) can be tackled by generalized Rayleigh-Ritz theorem and the solution is formed by the p eigenvectors of $D^{-1}L$ corresponding to the p smallest eigenvalues. LE is a non-linear method, it can't directly deal with new coming data.

He and Niyogi propose LPP to deal with out-of-sample problem (He and Niyogi 2003). LPP adopts the projection function $W \in \mathbb{R}^{d\times p}$ by replacing Y with XW. Then, the dimensionality reduction is performed as:

$$\min_{W} Tr((W^T X^T DXW)^{-1} W^T X^T LXW). \tag{2}$$

To avoid ill-posed problem, LPP can be reformulated as:

$$\min_{W} Tr((W^T (X^T DX + \alpha I)W)^{-1} W^T X^T LXW), \tag{3}$$

where α is the regularization parameter. LPP can be seen as the linearization extension of LE, it is a successful method and computationally efficient for training data as well as new data. However, there are also some disadvantages besides the performance degradation with non-linearly distributed data, Cai *et al.* point out that LPP is time consuming when calculating the generalized eigenvalue problem of the dense matrices $X^T LX$ and $X^T DX$ (Cai, He, and Han 2007).

Cai *et al.* then propose a different linearization extension method called SR (Cai, He, and Han 2007). SR first obtains the low dimension embedding Y^* by solving problem (1), then gets the projection matrix via solving a regression problem:

$$\min_{W} \|XW - Y^*\|_F^2. \tag{4}$$

Also, the regularized SR can be reformulated as:

$$\min_{W} \|XW - Y^*\|_F^2 + \alpha\|W\|_F^2. \tag{5}$$

Problem (5) can be efficiently solved by some well-studied algorithms, e.g. LSQR (Paige and Saunders 1982). SR is believed more efficient since it only needs to calculate generalized eigenvalue problem of the sparse matrices L and D (Cai, He, and Han 2007).

Equivalence between SR and LPP

Given similarity matrix which is doubly stochastic, we get that the degree matrix D is actually I. Substituting $L = D - A$ and $D = I$ into LPP formulation, i.e., problem (2), we get:

$$\max_{W} Tr((W^T X^T XW)^{-1} W^T X^T AXW). \tag{6}$$

Suppose that similarity matrix A is PSD with rank p, A can be decomposed by eigenvalue decomposition as:

$$A = F\Lambda F^T, \tag{7}$$

where $\Lambda \in \mathbb{R}^{n\times n} = diag(\lambda_1, \ldots, \lambda_n)$, $\lambda_1 \geq \cdots \geq \lambda_p > 0 = \lambda_{p+1} = \cdots = \lambda_n$ are the eigenvalues of A, and $F \in \mathbb{R}^{n\times n}$ is eigenvector matrix. It is easy to know that Eq. (7) can be rewritten as

$$A = F_p \Lambda_p F_p^T, \tag{8}$$

where $F_p \in \mathbb{R}^{n\times p}$ is the first p columns of F, and $\Lambda_p \in \mathbb{R}^{p\times p} = diag(\lambda_1, \ldots, \lambda_p)$. Then, substitute Eq. (8) into problem (6), we arrive at

$$\max_{W} Tr((W^T X^T XW)^{-1} W^T X^T F_p \Lambda_p F_p^T XW). \tag{9}$$

Problem (9) is nothing but the formulation of LPP with certain similarity matrix A.

On the other hand, given such similarity matrix A, the solution to LE, i.e. problem (1), is actually F_p, therefore, the formulation of SR, i.e. problem (4), can be rewritten as

$$\min_{W} \|XW - F_p\|_F^2. \tag{10}$$

Interestingly, although LPP and SR are two different spectral based dimensionality reduction methods, it can be verified that the solution space of problem (9) is exact equivalent to problem (10)'s, in other words, LPP is equivalent to SR if similarity matrix A is PSD, doubly stochastic and with rank p. Before presenting further proof, we first show an interesting observation of LPP as follows:

Lemma 1. *If $\hat{W}$ is the optimal solution to LPP, i.e. problem (2), $\hat{W}R$ is still an optimal solution, where $R \in \mathbb{R}^{p\times p}$ is an arbitrary invertible matrix.*

Proof. We directly substitute $\hat{W}R$ into problem (2) as

$$\begin{aligned} &Tr(((\hat{W}R)^T X^T DX\hat{W}R)^{-1}(\hat{W}R)^T X^T LX\hat{W}R) \\ =&Tr(R^{-1}(\hat{W}^T X^T DX\hat{W})^{-1}\hat{W}^T X^T LX\hat{W}R) \\ =&Tr((\hat{W}^T X^T DX\hat{W})^{-1}\hat{W}^T X^T LX\hat{W}) \end{aligned} \tag{11}$$

which means $\hat{W}R$ also makes problem (2) achieve optimal value. Thus, we complete the proof. □

Then, as a simple corollary, we can get that LE, i.e. problem (1), also has same property, we thus get following lemma for LE and SR as

Lemma 2. *Given F_p and $\hat{W}$, which are the optimal solutions to LE (i.e. problem (1)) and SR (i.e. problem (4)) respectively, F_pR and $\hat{W}R$ are also the optimal solution to LE and SR, respectively, where $R \in \mathbb{R}^{p\times p}$ is an arbitrary invertible matrix.*

Proof. One can easily check that if F_p is the optimal solution to LE, F_pR is still the optimal solution, the proof is similar to Lemma 1. We then just need to validate that $\hat{W}R$ is the optimal solution to following problem:

$$\min_{W} \|XW - F_pR\|_F^2. \tag{12}$$

To see this, note that $\hat{W}$ is the optimal solution to problem (4), we get the derivative of problem (4) as $X^T(X\hat{W} - F_p) = 0$, we then write the derivative of problem (12) and substituting $W = \hat{W}R$ as

$$X^T(X\hat{W}R - F_pR) = X^T(X\hat{W} - F_p)R = 0, \tag{13}$$

therefore, $\hat{W}R$ is the optimal solution to problem (12), which completes the proof. □

We then give a simple lemma as follows:

Lemma 3. *Given Λ is diagonal and positive definite, the column space of $A^{-1}B\Lambda B^T$ is exact same as that of $A^{-1}B$.*

Proof. On the one hand, it is easy to know that

$$\mathrm{Span}(A^{-1}B\Lambda B^T) \subseteq \mathrm{Span}(A^{-1}B), \tag{14}$$

where $\mathrm{Span}(M)$ denotes the space spanned by the column of the matrix M.

On the other hand, since Λ is positive definite, we can get

$$\begin{aligned} &\mathrm{rank}(A^{-1}B\Lambda B^T) = \mathrm{rank}(B\Lambda B^T) \\ =&\mathrm{rank}(B\Lambda^{\frac{1}{2}}(B\Lambda^{\frac{1}{2}})^T) = \mathrm{rank}(B\Lambda^{\frac{1}{2}}) = \mathrm{rank}(B) \\ =&\mathrm{rank}(A^{-1}B), \end{aligned} \tag{15}$$

where $\mathrm{rank}(M)$ denotes the rank of matrix M.

Combine Eq. (14) and Eq. (15), we know that these two spaces are exact same, thus we complete the proof. □

Based on Lemma 1, Lemma 2 and Lemma 3, we come up with following theorem:

Theorem 1. *If the symmetric similarity matrix A is doubly stochastic, PSD and with rank p, LPP is equivalent to SR.*

Proof. As mentioned above, with such specific similarity matrix, problem (9) and problem (10) are actually LPP and SR, respectively. According to Lemma 1 and Lemma 2, the solution spaces to problem (9) and problem (10) are the column space of $(X^TX)^{-1}X^TF_p\Lambda_pF_p^TX$ and $(X^TX)^{-1}X^TF_p$, respectively. Moreover, based on Lemma 3, we know that the column spaces of these two matrix are same, in other words, the solution space of LPP is same as that of SR, we then complete the proof. □

We further show that regularized LPP, i.e. problem (3), is equivalent to regularized SR, i.e. problem (5), by following theorem:

Theorem 2. *If similarity matrix A is symmetric, doubly stochastic, PSD and with rank p, regularized LPP, i.e. problem (3), is equivalent to regularized SR, i.e. problem (5).*

The proof is similar to Theorem 1, and we omit it here.

We next show some important significances of Theorem 1 as well as Theorem 2 as follows, for simplicity, we take Theorem 1 as an example:

1. Although LPP and SR are different methods, with certain similarity matrix, SR does make sense in the framework of LPP, and vice versa.
2. Theorem 1 gives a new approach to tackle LPP problem with certain similarity matrix, i.e. through solving SR problem.
3. Theorem 1 gives a new approach to tackle SR problem with certain similarity matrix, i.e. through solving LPP problem.

The second point is particularly valuable, as pointed by Cai *et al.* (Cai, He, and Han 2007), LPP suffers from high computational cost, they then propose a more efficient method, i.e. SR. According to Theorem 1, instead of directly tackling LPP problem, we can now efficiently tackle specific SR problem to get the solution of LPP's. Moreover, as pointed by (Zass and Shashua 2006)(Wang, Nie, and Huang 2016), doubly stochastic similarity matrix always results in promising performance. Therefore, it is reasonably to construct the required similarity matrix in Theorem 1 to get both high efficiency and high performance.

Unsupervised Large Graph Embedding

In this section, we show the proposed ULGE. ULGE starts with similar idea as LPP, however, according to Theorem 1, it is solved by a SR liked algorithm.

Similarity Matrix Construction with Anchor-based Strategy

Theorem 1 requires that similarity matrix A must be doubly stochastic, PSD and with rank p, however, conventional spectral based methods always adopt k-nearest neighbor approach to construct similarity matrix, which is not only time consuming but also results in a almost full-rank and non doubly stochastic similarity matrix. However, it is difficult to construct doubly stochastic, PSD and rank-p similarity matrix simultaneously, we thus propose a two-step approach, for detail, we first construct doubly stochastic and PSD similarity matrix, and then use it to construct rank-p similarity matrix.

Anchor Generation Following recent studies on scalable semi-supervised learning (Liu, He, and Chang 2010), we adopt an efficient method to construct such similarity matrix, i.e. anchor-based strategy. In general, anchor-based strategy first seeks m anchors, where $m \ll n$, and then calculates the distance between anchors and original samples.

The most important step of anchor-based strategy is anchor generation, and there are mainly two methods, i.e. random selection and k-means generation. It is efficient to generate anchors by simply random sampling, nonetheless, we still

prefer to use k-means to generate more representative anchors for better performance. Note that, k-means may cost too much time if the data is too large, to the best of our knowledge, two simple strategies can be adopted to speed up the procedure, i.e. early stopping the iteration (Chen and Cai 2011) and performing down-sampling as preprocessing, we adopt down-sampling strategy in this paper.

We would also like to underline that, for k-means anchor generation, both of these two speed up strategies can not guarantee the quality of the generated anchors, and one of our future work is trying to generate representative anchors with a novel balanced k-means based hierarchical k-means algorithm, which is expected to have high performance as well as low computational complexity. For detail, we first design a balanced k-means algorithm which can separate the data into two clusters with same number of samples, and then hierarchically perform the balanced k-means algorithm on the data to get the representative anchors. Denoting t as the number of iterations, the computational complexity of this algorithm is $O(nd\log(m)t)$, which has great advantage over k-means method whose complexity is $O(ndmt)$. Note that, such anchor generation method can be easily applied to accelerate other graph based learning methods, e.g. hashing (Li, Hu, and Nie 2017), clustering (Ng, Jordan, and Weiss 2001), semi-supervised learning (Zhou et al. 2003; Zhu 2008), dimensionality reduction (Nie et al. 2011), RBF networks (Schwenker, Kestler, and Palm 2001), etc.

Anchor-based Similarity Matrix Let $U_{\langle i\rangle}$ denote the set of k-nearest anchors for the i-th sample, where $U \in \mathbb{R}^{m\times d}$ is the set of whole anchors. Conventional methods usually use kernel based neighbor assignment strategy, e.g. Gaussian kernel $K_t(x_i,u_j) = exp(-\|x_i-u_j\|_2^2/\tau^2)$, but kernel based methods always bring extra parameters, e.g. bandwidth τ. To avoid this, we adopt a parameter-free yet effective neighbor assignment strategy (Nie et al. 2016). The neighbor assignment for the i-th sample can be seen as solving following problem (Nie et al. 2016)

$$\min_{z_i^T\mathbf{1}=1, z_i\geq\mathbf{0}} \sum_{j=1}^{m} h(x_i,u_j)z_{ij} + \gamma\sum_{j=1}^{m} z_{ij}^2, \tag{16}$$

where $Z \in \mathbb{R}^{n\times m}$ denotes the similarity between the i-th sample and the j-th anchor, $h(x_i,u_j)$ is the distance between the i-th sample and its j-th nearest anchor, to keep it simple, we define $h(x_i,u_j) = \|x_i-u_j\|_2^2$, which is the square of Euclidean distance. Follow (Nie et al. 2016), γ can be set as $\gamma = \frac{k}{2}h(i,k+1) - \frac{1}{2}\sum_{j=1}^{k} h(i,j)$. The solution to problem (16) is

$$z_{ij} = \frac{h(x_i,u_{k+1}) - h(x_i,u_j)}{\sum_{j'=1}^{k}\left(h(x_i,u_{k+1}) - h(x_i,u_{j'})\right)}. \tag{17}$$

For detail deviation, see (Nie et al. 2016).

After we obtain the matrix Z, similarity matrix A then can be obtained by (Liu, He, and Chang 2010):

$$A = Z\Delta^{-1}Z^T, \tag{18}$$

where $\Delta \in \mathbb{R}^{m\times m}$ is a diagonal matrix and the j-th entry is defined as $\sum_{i=1}^{n} z_{ij}$. One can simply check that matrix A is symmetric, PSD, and doubly stochastic(Liu, He, and Chang 2010).

Low Rank Approximation of Anchor-based Similarity Matrix However, the rank of A obtained by solving problem (18) is always larger than p in most cases, we thus seek a rank-p approximation of A as

$$\min_{\tilde{A}} \|\tilde{A}-A\|_F, \quad s.t. \quad rank(\tilde{A}) = p, \tag{19}$$

where $\tilde{A} \in \mathbb{R}^{n\times n}$. The optimal solution $\tilde{A}^*$ to problem (19) is $\tilde{A}^* = F_p\Lambda_p F_p^T$ according to Eckart-Young-Mirsky theorem (Eckart and Young 1936). The approximated similarity matrix $\tilde{A}^*$ is obviously PSD, symmetric and with rank p, we now show that it is still doubly stochastic as follows:

Lemma 4. *Given matrix $A \in \mathbb{R}^{n\times n}$ is symmetric, PSD and doubly stochastic, the rank-p approximation $\tilde{A}_p$ is still doubly stochastic, where $1 \leq p \leq n$.*

Proof. According to Gerschgorin's disk theorem, since A is PSD and doubly stochastic, the maximum eigenvalue of A is 1, and the corresponding eigenvector is $\frac{\mathbf{1}}{\sqrt{\mathbf{n}}}$. Then, we perform eigenvalue decomposition of A as

$$A = \lambda_1 f_1 f_1^T + \lambda_2 f_2 f_2^T + \ldots + \lambda_n f_n f_n^T, \tag{20}$$

where $0 \leq \lambda_i \leq 1$ is the i-th largest eigenvalue and f_i is the corresponding eigenvector. According to Eckart-Young-Mirsky theorem (Eckart and Young 1936), rank-p approximation of A is constructed as

$$\tilde{A}_p = \lambda_1 f_1 f_1^T + \lambda_2 f_2 f_2^T + \ldots + \lambda_p f_p f_p^T \tag{21}$$

Note that A is symmetric, according to the orthogonality of the eigenvector, we get

$$\begin{aligned}\tilde{A}_p\mathbf{1} &= \lambda_1 f_1 f_1^T\mathbf{1} + \lambda_2 f_2 f_2^T\mathbf{1} + \ldots + \lambda_p f_p f_p^T\mathbf{1} \\ &= \frac{\mathbf{1}}{\sqrt{\mathbf{n}}}(\frac{\mathbf{1}}{\sqrt{\mathbf{n}}})^T\mathbf{1} + \mathbf{0} + \ldots + \mathbf{0} \\ &= \mathbf{1}\end{aligned} \tag{22}$$

Similarly, we can get $\mathbf{1}^T\tilde{A}_p = \mathbf{1}^T$. That is, $\tilde{A}_p$ is obviously still doubly stochastic, which we complete the proof. □

The optimal solution to problem (19), i.e. $\tilde{A}^*$, can be directly obtained by the eigenvalue decomposition on A, however, note that Eq. (18) can be rewritten as

$$A = BB^T, \tag{23}$$

where $B = Z\Delta^{-\frac{1}{2}} \in \mathbb{R}^{n\times m}$, we then can perform Singular Value Decomposition (SVD) on B instead of performing eigenvalue decomposition on A to get the solution, the time complexity is $O(nmk)$, where k is the number of nearest neighbors. At last, we obtain the specific similarity matrix which is required by Theorem 1.

Spectral Analysis with Anchor-based Graph

As mentioned, ULGE starts with similar idea as LPP. Since the similarity matrix $\hat{A}^*$ exactly meets the conditions of Theorem 2, ULGE can then be tackled by a simple regression problem as

$$\min_W \|XW - F_p\|_F^2 + \alpha\|W\|_F^2. \quad (24)$$

To avoid ill-posed problem, we adopt regularization term. We summarize the detail algorithm of ULGE in Algorithm 1.

Algorithm 1 Large Graph Embedding Dimensionality Reduction

Input: Data matrix $X \in \mathbb{R}^{n\times d}$, projection dimension p, number of anchors m, regularization parameter α

1. Generate m anchors by perform k-means or simply random sampling.
2. Obtain the similarity matrix A by tackle problem (18) with anchor-based graph.
3. Obtain F_p by performing SVD on matrix B, where $B = Z\Delta^{-\frac{1}{2}}$.
4. Compute the projection matrix W by solving problem (24).

Output: Projection matrix $W \in \mathbb{R}^{d\times p}$.

Computational Complexity Analysis

Given a data matrix $X \in \mathbb{R}^{n\times d}$, the computational complexity of ULGE is divided into 4 parts.

1. We need $O(ndmt)$ to obtain m anchors by k-means, also we can randomly select anchors which has a time complexity $O(1)$.
2. We need $O(ndm + nm\log(m))$ to construct graph by anchor-based approach.
3. We need $O(nmk)$ to obtain F_p by solving problem (19), where k is the number of nearest neighbors.
4. We need $O(dnp)$ to obtain projection matrix W by solving problem (24).

Considering that $m \ll n$ and $d \ll n$ for large scale data, the overall computational complexity is $O(ndm)$. Compared to conventional spectral based methods which need at least $O(n^2d)$, ULGE has great computational advantage especially on large scale data sets.

Table 1: Data Set Description

Data Set	Samples	Features	Classes
USPS	9298	256	10
Protein	24387	357	3
Connect-4	67557	126	3
MNIST	70000	784	10
SensIT	98528	100	3

Experiments

In this section, we experimentally demonstrate the efficiency and effectiveness of the proposed method on 5 benchmark large scale data sets, and then show several useful analysis.

Data Sets

We conduct experiments on 5 different public available data sets downloaded from the LibSVM data sets page[1], UCI machine learning repository [2], and Deng Cai's page [3]. The data sets includes handwritten digit (e.g. MNIST and USPS), types of moving vehicles (e.g. SensIT), molecular biology (e.g. Protein), and connect-4 game (e.g. Connect-4). The detail of the data sets are summarized in Table 1 .

Comparison Methods

To validate the advantage of ULGE, we compare it with several conventional spectral based unsupervised dimensional reduction methods. e.g. LE, LPP and SR. We also compare the performance of ULGE with different anchor selection strategies, i.e. ULGE-K (ULGE with k-means anchor generation) and ULGE-R (ULGE with random anchor selection). We perform k-means with original data as Baseline to validate the effectiveness of all the dimensionality reduction methods.

We use same Gaussian kernel for all LE, LPP and SR, and the reduced dimension is set as number of class of the data set. We use 5-nearest neighbor to construct graph for all the methods. The regularization parameter α is default set as 0.01 in both SR and ULGE. For parameter m, i.e. the number of anchors, used in ULGE, we empirically set $m = 1000$. To speed up ULGE-K, we suggest to perform down sampling to generate anchors, and in this paper, the decimation factor is set as 10 for all data sets except USPS which is set as 3.

Evaluation Metric

For all the methods, we run 10 times k-means on the reduced subspace, then evaluate the clustering result by ACCuracy (ACC). We record the mean results as well as the running time of all methods. All the codes in the experiments are implemented in MATLAB R2015b , and run on a Windows 10 machine with 3.20 GHz i5-3470 CPU, 16 GB main memory.

Clustering Results

The performance of dimensionality reduction methods evaluated by ACC is reported in Table 2, and the running time is reported in Table 3. We present several interesting points as follows:

Compared to other spectral based methods, the proposed ULGE-K and ULGE-R achieve competitive performance for almost all the data sets. As mentioned above, anchor-based strategy is not only effective but also results in doubly stochastic similarity matrix, ULGE-K thus exceeds other non-linear and linear dimensionality reduction methods in most cases. ULGE-R also achieves pretty good performance though it selects anchors randomly. As the number of anchors is set as 1000 in this paper, the performance of both ULGE-K and ULGE-R are expected to get better performance with more anchors, we thus suggest increasing the number of anchors in real life application.

[1] https://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/

[2] http://archive.ics.uci.edu/ml/

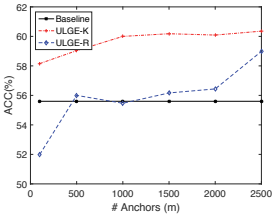
[3] http://www.cad.zju.edu.cn/home/dengcai/Data/data.html

Table 2: Clustering results on 5 large scale data sets. (Top 2 rank methods are highlighted in bold.)(%)

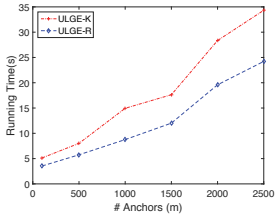
Data Set	Baseline	LE	LPP	SR	ULGE-K	ULGE-R
USPS	66.9	**68.9**	67.0	68.1	**69.3**	63.9
Protein	42.7	43.9	44.0	**44.0**	**44.3**	43.7
Connect-4	37.4	44.3	39.6	**48.9**	**47.5**	45.9
MNIST	55.6	**68.4**	51.3	57.9	**60.7**	54.9
SensIT	**68.9**	61.7	**69.3**	65.2	67.0	65.6

Table 3: Running time on 5 large scale data sets. (Top 2 rank methods are highlighted in bold.) (s)

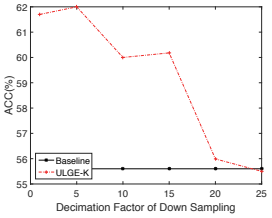
Data Set	LE	LPP	SR	ULGE-K	ULGE-R
USPS	36.4	32.7	38.7	**21.5**	**7.2**
Protein	73.3	64.0	82.8	**14.5**	**10.2**
Connect-4	398.7	322.9	400.1	**30.2**	**24.1**
MNIST	242.6	159.4	249.5	**14.7**	**8.8**
SensIT	1492.1	1489.7	1505.1	**91.5**	**47.6**



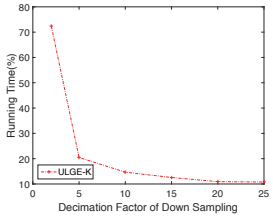
(a) ACC VS. # of anchors.



(b) Running time VS. # of anchors.



(c) ACC VS. decimation factor of down sampling.



(d) Running time VS. decimation factor of down sampling.

Figure 1: ACC and running time with different parameters.

Considering the running time, the proposed ULGE-K and ULGE-R achieve significant improvements, especially for large scale data set. For the largest data set SensIT which contains 98528 samples, ULGE-K and ULGE-R only need 91.5 and 47.6 seconds, respectively, which is 16 and 31 times faster than the third fastest method LPP. As the growth of scale of data set, ULGE-K and ULGE-R surely become more and more superior. The running time of k-means can be obtained by the difference between ULGE-K and ULGE-R, we see that ULGE-K is greatly limited by k-means even with down sampling. Predictably, ULGE-R will be the only choice in extreme situations if it is too time consuming to perform k-means for ULGE-K.

Parameters Selection

We only study the most important parameters, i.e. m (number of anchors) and decimation factor of down sampling (only for ULGE-K), the experiments are conducted on MNIST data set.

Figure 1(a) shows that the more anchors we selected, the better performance we will get. Moreover, the curve of ULGE-R tends to be closer to ULGE-R's, which means that k-means step is meaningless if we select too many anchors. However, increasing number of anchors also make the time cost increase, e.g., we need 14.7 seconds for 1000 anchors and 34.3 seconds for 2500 anchors for ULGE-K. As the decimation factor increasing, the performance as well as time cost drops much as show in Figure 1(c) and Figure 1(d). To be specific, we need 72.3 seconds for a decimation factor of 2, and 14.7 seconds for a decimation factor of 10. However, if we still increase the decimation factor, the time cost drops little.

Conclusions

In this paper, we propose an interesting theorem about the relation between LPP and SR, which are two pretty different linear dimensionality reduction methods. Inspired by the theorem, we then propose an efficient and effective dimensionality reduction method, called Unsupervised Large Graph Embedding (ULGE). ULGE starts with similar idea as LPP, it adopts anchor-based strategy to construct PSD and doubly stochastic similarity matrix, then performs low-rank approximation to get rank-p similarity matrix. The overall computational complexity is $O(ndm)$, which is a significant improvement compared to conventional methods which need $O(n^2d)$ at least. Extensive experiments conducted on 5 large scale data sets demonstrate the efficiency and effectiveness of ULGE. One of our future work is designing an effective anchor generation algorithm with extremely low computational complexity to replace the time consuming k-means method for ULGE-K.

References

Belkin, M., and Niyogi, P. 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, 585–591.

Cai, D., and Chen, X. 2015. Large scale spectral clustering via landmark-based sparse representation. *IEEE T. Cybernetics* 45(8):1669–1680.

Cai, D.; He, X.; and Han, J. 2007. Spectral regression: a unified subspace learning framework for content-based image retrieval. In *Proceedings of the 15th International Conference on Multimedia*, 403–412.

Cai, D. 2015. Compressed spectral regression for efficient nonlinear dimensionality reduction. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 3359–3365.

Chen, X., and Cai, D. 2011. Large scale spectral clustering with landmark-based representation. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Chung, F. R. K. 1997. *Spectral Graph Theory*. American Mathematical Society.

Eckart, C., and Young, G. 1936. The approximation of one matrix by another of lower rank. *Psychometrika In Psychometrika* 1(3):211–218.

He, X., and Niyogi, P. 2003. Locality preserving projections. In *Advances in Neural Information Processing Systems 16*, 153–160.

Li, Y.; Nie, F.; Huang, H.; and Huang, J. 2015. Large-scale multi-view spectral clustering via bipartite graph. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2750–2756.

Li, X.; Hu, D.; and Nie, F. 2017. Large graph hashing with spectral rotation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*.

Liu, W.; He, J.; and Chang, S. 2010. Large graph construction for scalable semi-supervised learning. In *Proceedings of the 27th International Conference on Machine Learning*, 679–686.

Ng, A. Y.; Jordan, M. I.; and Weiss, Y. 2001. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, 849–856.

Nie, F.; Xu, D.; Tsang, I. W.; and Zhang, C. 2010. Flexible manifold embedding: A framework for semi-supervised and unsupervised dimension reduction. *IEEE Transactions on Image Processing* 19(7):1921–1932.

Nie, F.; Xu, D.; Li, X.; and Xiang, S. 2011. Semisupervised dimensionality reduction and classification through virtual label regression. *IEEE Trans. Systems, Man, and Cybernetics, Part B* 41(3):675–685.

Nie, F.; Wang, X.; Jordan, M. I.; and Huang, H. 2016. The constraint laplacian rank algorithm for graph-based clustering. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Paige, C. C., and Saunders, M. A. 1982. LSQR: an algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Softw.* 8(1):43–71.

Schwenker, F.; Kestler, H. A.; and Palm, G. 2001. Three learning phases for radial-basis-function networks. *Neural Networks* 14(4-5):439–458.

T, R. S., and K, S. L. 2000. Nonlinear dimensionality reduction by locally linear embedding. *Science* 290(5):2323–2326.

Tenenbaum, J. B. 1997. Mapping a manifold of perceptual observations. In *Advances in Neural Information Processing Systems 10*, 682–688.

Wang, H.; Yan, S.; Xu, D.; Tang, X.; and Huang, T. S. 2007. Trace ratio vs. ratio trace for dimensionality reduction. In *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Wang, X.; Nie, F.; and Huang, H. 2016. Structured doubly stochastic matrix for graph based clustering: Structured doubly stochastic matrix. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1245–1254.

Yan, S.; Xu, D.; Zhang, B.; Zhang, H.; Yang, Q.; and Lin, S. 2007. Graph embedding and extensions: A general framework for dimensionality reduction. *IEEE Trans. Pattern Anal. Mach. Intell.* 29(1):40–51.

Zass, R., and Shashua, A. 2006. Doubly stochastic normalization for spectral clustering. In *Advances in Neural Information Processing Systems*, 1569–1576.

Zhou, D.; Weston, J.; Gretton, A.; Bousquet, O.; and Schölkopf, B. 2003. Ranking on data manifolds. In *Advances in Neural Information Processing Systems 16*, 169–176.

Zhu, X. 2008. Semi-supervised learning literature survey. *Computer Science* 37(1):63–77.